```
#INTEGRATION WITH EULER TECHNIQUE

from visual import *

#G is gravitational constant in N-m^2/kg^2, Re is earth radius in m
#Me is earth mass in kg
#radially falling object with acceleration -GMe/r^2

G=6.67e-11
Re=6.37e6
Me=5.98e24

#Evaluate position and velocity vs time, h and v vs t for object falling
from h0
#h0 and h in m, v in m/s, set initial conditions
#define h, v, t, and a as lists each with one initial element

h0=2.5e6
h=[h0+Re]
v=[0]
t=[0]
a=[-G*Me/h[0]**2]
#print a[0]

#  dt is the time increment for Euler's eqn, units seconds
#repeat calculation with variable value of dt and evaluate change in t and
v

dt=100.0

icount=0

print "      time    height    velocity    accel'n"  #add this once the
program works

def Eulerstep(y,h,f):
    return y[-1]+f[-1]*h

def EulerInt(y,v,a,h):
    y.append(Eulerstep(y,h,v))  #  this is h0 +v0*dt = h1, etc.
    v.append(Eulerstep(v,h,a))  #  this is v0 +a0*dt = v1, etc.
    a.append(-G*Me/y[-1]**2)  #  this is -GMe/h1^2 = a1, etc.
    t.append(t[-1]+dt)

while h[-1]>Re:

   print icount
   icount = icount+1

   #Euler method----Note the order; would give different results if changed
   EulerInt(h,v,a,dt)
#   h.append(Eulerstep(h,dt,v))  #  this is h0 +v0*dt = h1, etc.
#   v.append(Eulerstep(v,dt,a))  #  this is v0 +a0*dt = v1, etc.
#   a.append(-G*Me/h[-1]**2)  #  this is -GMe/h1^2 = a1, etc.
#   t.append(t[-1]+dt)        #  t1 = t0 + dt, etc.

   print "%10.0f %10.4E %10.4E %10.4E" % (t[-1], h[-1], v[-1], a[-1])

print " "
```

```
print "the array length and counter end values are %d and %d" % (len(h),
icount)
print "the value of dt is %10.2f" % (dt)
print "the time at which Re was reached "
print "%10.1f" % (t[-2] + (t[-1]-t[-2])*(h[-2]-Re)/(h[-2]-h[-1]))
print "the value of v at Re"
print "%10.4E" %  (v[-2] + (v[-1]-v[-2])*(h[-2]-Re)/(h[-2]-h[-1]))

#analytical result for h0=2.5e6 is 5.94e3 m/s
#analytical result for h0=2.5e5 is 2.17e3 m/s

#########################################################################

#INTEGRATION WITH RK2

from visual.graph import *

#G is gravitational constant in N-m^2/kg^2, Re is earth radius in m
#Me is earth mass in kg
#radially falling object with acceleration -GMe/r^2

G=6.67e-11
Re=6.37e6
Me=5.98e24

#Evaluate position and velocity vs time, h and v vs t for object falling
from h0
#h0 and h in m, v in m/s, set initial conditions
#define h, v, t, and a as lists each with one initial element

h0=2.5e6
h=[h0+Re]
v=[0]
t=[0]
a=[-G*Me/h[0]**2]
#print a[0]

#  dt is the time increment for Euler's eqn, units seconds
#repeat calculation with variable value of dt and evaluate change in t and
v

dt=100.0

icount=0

print "      time    height    velocity    accel'n"  #add this once the
program works

def Eulerstep(y,h,f):
    return y[-1]+f[-1]*h

def RK2Int(y,v,a,h):
    yEu=Eulerstep(y,h,v)
    vEu=Eulerstep(v,h,a)
    aEu=-G*Me/yEu**2
    y.append(y[-1]+0.5*(v[-1]+vEu)*h)  #  this is h0 +v0*dt = h1, etc.
    v.append(v[-1]+0.5*(a[-1]+aEu)*h)  #  this is v0 +a0*dt = v1, etc.
    a.append(-G*Me/y[-1]**2)  #  this is -GMe/h1^2 = a1, etc.
    t.append(t[-1]+dt)
```

```
while h[-1]>Re:

   print icount
   icount = icount+1


   RK2Int(h,v,a,dt)
#   h.append(Eulerstep(h,dt,v))  #  this is h0 +v0*dt = h1, etc.
#   v.append(Eulerstep(v,dt,a))  #  this is v0 +a0*dt = v1, etc.
#   a.append(-G*Me/h[-1]**2)  #  this is -GMe/h1^2 = a1, etc.
#   t.append(t[-1]+dt)        #  t1 = t0 + dt, etc.

   print "%10.0f %10.4E %10.4E %10.4E" % (t[-1], h[-1], v[-1], a[-1])

print " "
print "the array length and counter end values are %d and %d" % (len(h),
icount)
print "the value of dt is %10.2f" % (dt)
print "the time at which Re was reached "
print "%10.1f" % (t[-2] + (t[-1]-t[-2])*(h[-2]-Re)/(h[-2]-h[-1]))
print "the value of v at Re"
print "%10.4E" %  (v[-2] + (v[-1]-v[-2])*(h[-2]-Re)/(h[-2]-h[-1]))

#analytical result for h0=2.5e6 is 5.94e3 m/s
#analytical result for h0=2.5e5 is 2.17e3 m/s
```